ON A BIOBJECTIVE FLOW PROBLEM IN NETWORKS

By

SAMPREET S. MANGALVEDHE

Bachelor of Engineering
University of Mumbai
Mumbai, India
2012

Master of Science in Industrial Engineering &
Management
Oklahoma State University
Stillwater, OK
2016

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2016

ON A BIOBJECTIVE FLOW PROBLEM IN NETWORKS

Thesis approved:

Dr. Balabhaskar Balasundaram

Thesis Advisor

Dr. Austin Buchanan

Dr. Farzad Yousefian

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Balasundaram Balabhaskar for his support and patience. He helped me find the right direction for my research and guided me through his helpful inputs. I would also like to thank Dr. Austin Buchanan and Dr. Farzad Yousefian for their active participation and valuable comments. Their feedback steered my thesis further and showed the room for improvement.

Finally, I would thank my parents, brother and friends for providing me continuous support and encouragement. This accomplishment would not have been possible without them.

Disclaimer: Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: Sampreet Sudheer Mangalvedhe Date of Degree: December, 2016

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: ON A BIOBJECTIVE FLOW PROBLEM IN NETWORKS

Candidate for the Degree of Master of Science

Major Field: Industrial Engineering & Management

Abstract: Supply chain disruptions not only impact the regular operations, but can also affect the reputation of an organization. In this thesis, we considered a capacitated network that transported products from the source node to the sink node based on an operating plan. This network underwent a disruption, which led to arc closures. The problem of interest is to move the products from the source to the sink with minimal deviation from the original operating plan, and also transport a sufficiently large amount of the products. We proposed optimization models to minimize the dissimilarity between two operating plans and also transport a sufficient value of flow to the sink. These models are motivated by lexicographic goal programming philosophy. Further, we implemented these models to understand the scalability of these models. The solver parameters were tuned to enhance the computational performance of our models. We also developed a visualization aid for the end-user. The visualization was designed to help the user understand the merits of our models. We also conducted a visualization experiment to understand the impact of number of disrupted arcs on dissimilarity.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Network models are pervasive. They offer an intuitive way for representing a complex system of interconnected components. We find their application in many settings; power transmission, transportation, communication systems are some of the familiar applications. Many of the real-world problems are modeled as networks because they provide a structure that facilitates the development of algorithms.

In the maximum flow problem, we intend to send as much flow as possible from a source node to a sink node in a capacitated network without violating capacity and flow conservation constraints [1]. There are numerous instances where this problem arises [1] and hence it is extensively studied. This thesis focuses on a biobjective variant of the maximum flow problem, where one objective is to minimize the dissimilarity between two solutions and the other objective is to send a sufficient amount of flow to the sink.

1.1 Preliminaries

1.1.1 The maximum flow problem

Let $G = (N, A)$ be a finite, directed network, with N as the set of nodes and A as the set of arcs. Let s and t be two special nodes that denote a source and a sink respectively in the network G. Let x_{ij} denote the amount of flow through an arc (i, j). The capacity of an arc (i, j) is defined as the maximum amount of flow that can pass through arc (i, j). Let us denote the capacity of arc (i, j) by u_{ij}. A conservation law

must be satisfied at every node except s and t. That is, the flow that goes out of a node i must equal the flow that comes into node i. Given a capacitated network $G = (N, A)$ and $u_{ij} \geq 0 \ \forall (i,j) \in A$, the maximum flow problem seeks to send a maximum amount of flow from the source node s to the sink node t while obeying all arc capacities and flow conservation at all nodes (except the source and the sink). The maximum flow problem can be formulated as follows:

$$\max \quad v \tag{1.1}$$

subject to:

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v, & \textbf{if} \quad i = s \\ 0, & \forall i \in N \setminus \{s,t\} \\ -v, & \textbf{if} \quad i = t \end{cases} \tag{1.2}$$

$$x_{ij} \leq u_{ij}, \qquad \forall (i,j) \in A \tag{1.3}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in A \tag{1.4}$$

In the above formulation, the constraints (1.2) enforce the conservation law and hence these constraints are called the flow conservation constraints. Constraints (1.3) and (1.4) define the upper and lower bounds on x_{ij}. Constraints (1.3) are called the capacity constraints.

Any assignment of nonnegative values to $x_{ij} \ \forall (i,j) \in A$ that satisfies the flow conservation constraints and the capacity constraints is called a feasible flow or simply, flow, and v is its value. A maximum flow corresponding to the maximum flow value v^*, is denoted by x^*. In this thesis, we assume that there is an s-t path in G with positive "bottleneck" capacity, i.e. every arc (i,j) on this path has $u_{ij} > 0$. We further assume that G does not contain an s-t path of ∞ bottleneck capacity, i.e. on every s-t

path there is some arc (i,j) with $u_{ij} < \infty$. Under these assumptions the formulations (1.1) - (1.4) has an optimal solution that is also integral due to total unimodularity of the constraint matrix [1]. Before we present some key theorems, we introduce additional terminology.

Definition 1 (Cut) *Given a partition of the node set N into S and $\bar{S}$, the cut of S denoted by $[S, \bar{S}]$, is defined as the set of arcs with one end-point in S and the other in $\bar{S}$.*

Definition 2 (s-t cut) *A source-sink cut, or $s-t$ cut for short, is a cut $[S, \bar{S}]$ such that $s \in S$ and $t \in \bar{S}$.*

We are interested only in $s-t$ cuts and henceforth we will simply refer to it as a *cut*.

Definition 3 (Capacity of cut) *An arc $(i, j) \in A$ is said to be a forward arc if $i \in S$ and $j \in \bar{S}$. The capacity of a cut $[S, \bar{S}]$ is the sum of the forward arc capacities. We use $(S, \bar{S})$ to denote the forward arcs in $[S, \bar{S}]$. Therefore,*

$$u[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} u_{ij}.$$

Definition 4 (Minimum cut) *A minimum cut is an $s-t$ cut $[S, \bar{S}]$ that has the minimum capacity among all $s-t$ cuts.*

Theorem 1 (Maximum Flow Minimum Cut Theorem) *The maximum value of a flow is equal to the minimum capacity among all source-sink cuts. A flow x^* with value v^* is the maximum flow if and only if there exists some $s-t$ cut $[S, \bar{S}]$ such that $v^* = u[S, \bar{S}]$.*

Theorem 1 establishes the strong duality between the maximum flow problem and the minimum cut problem. The next theorem provides another optimality condition/characterization for a flow in G, which is based on the notion of a residual network.

Definition 5 (Residual network) *The residual capacity on a particular arc (i,j) is made up of two components: (i) unused capacity $u_{ij} - x_{ij}$, and (ii) the flow on the opposite arc (j, i) given by x_{ji} (if it exists), canceling which has the same effect as increasing the flow on (i, j). Formally, $r_{ij} = u_{ij} - x_{ij} + x_{ji}$ for each arc $(i, j) \in A$. Given a capacitated network $G = (N, A)$, and a flow $x \in \mathbb{R}_+^{|A|}$, a residual network with respect to x denoted by $G(x)$ only contains those arcs of A that have positive residual capacity.*

Theorem 2 (Augmenting Path Theorem) *A flow x^* is maximum if and only if the residual network $G(x^*)$ does not contain a directed path from the source to the sink.*

1.1.2 Multi-objective optimization

Many real-world problems deal with multiple conflicting objectives. In such cases, describing an 'optimal' solution is nontrivial. One can have a solution that is optimal with respect to one objective but might turn out to be inferior for another objective. We explain the following terminologies that facilitate a better understanding of 'optimal' solutions in multi-objective optimization problems [2, 3].

Definition 6 (Efficient solution) *A feasible solution y' is called efficient if there exists no other feasible solution that is at least as good as y' with respect to all objectives and strictly better with respect to at least one objective. Such a solution is also called a Pareto optimal solution.*

Definition 7 (Efficient frontier) *The set of all efficient solutions is called an efficient frontier or the Pareto optimal set.*

Definition 8 (Ideal point) *A solution which is efficient with respect to all objectives is called an ideal point.*

1.2 Problem statement and motivation

Consider a capacitated pipeline network modeled as $G^o = (N, A^o)$ which transports petroleum products from various sources to different cities. Based on an operating plan, there are certain pipelines (i.e., arcs) that carry flow while certain arcs are not utilized. Let us say this network is disrupted for some reason (e.g., maintenance, natural hazards), which has led to pipeline closures. With this reconstructed network $G = (N, A)$, where $A \subset A^o$, the company would still wish to move products while deviating minimally from its current operating plan for a variety of reasons (e.g., transportation risk, time, cost, etc). This gives rise to two objectives: i) transporting a sufficiently large quantity of petroleum to customers, and ii) minimal deviation of arc flows from the current operating plan.

The aforementioned situation motivated us to consider a biobjective network flow problem, where one would like to minimize a dissimilarity metric between two flows and still send the sufficient amount of flow to the sink. The following definition will aid us in better describing the problem.

Definition 9 (Support of flow, S(x)) *Given a flow x on a network $G = (N, A)$, the support of x is defined as the arc set in which each arc has a positive flow on them. Formally, it is defined as, $S(x) = \{(i, j) \in A \mid x_{ij} > 0\}$.*

Consider a network $G^o = (N, A^o)$ carrying a flow x^o from s to t. Suppose this network, after some arc closures, is represented by $G = (N, A)$ where $A \subset A^o$. Suppose x denotes a feasible $s - t$ flow in G. We can quantify the dissimilarity between x^o and x in two different ways as described next.

Support dissimilarity: The symmetric difference between $S(x)$ and $S(x^o)$ is made up of two sets: (i) the set of arcs that belong to the support of x but not to the support of x^o, i.e., $S(x) \setminus S(x^o)$, and (ii) the set of arcs that belong to the support of x^o but not to the support of x, given by $S(x^o) \setminus S(x)$. Using these sets, we define

the support dissimilarity as the cardinality of the symmetric difference between $S(x)$ and $S(x^o)$, given by:

$$\mid S(x) \triangle S(x^o) \mid = \mid S(x) \setminus S(x^o) \mid + \mid S(x^o) \setminus S(x) \mid .$$

Since $A \subset A^o$, we can classify A into two sets: (i) $A \setminus S(x^o)$, and (ii) $A \cap S(x^o)$. Finding a flow x that minimizes $\mid S(x) \triangle S(x^o) \mid$ discourages flow along the arcs in $A \setminus S(x^o)$, and encourages flow along the arcs in $A \cap S(x^o)$, regardless of the amount of flow on these arcs.

Total absolute flow deviation: We measure the sum of absolute differences between x_{ij} and x^o_{ij} for each arc $(i,j) \in A$. Finding a flow x that minimizes the sum of absolute differences between x_{ij} and x^o_{ij} for each arc $(i,j) \in A$ encourages the arcs to carry the flow similar in magnitude to the flow in G^o. We denote this metric by $d(x, x^o)$, defined as:

$$d(x, x^o) = \sum_{(i,j) \in A} \mid x_{ij} - x^o_{ij} \mid .$$

Biobjective Flow Problem: Consider a capacitated network $G^o = (N, A^o)$ carrying an $s-t$ flow x^o, and a network $G = (N, A)$, where $A \subset A^o$. The problem of interest is to find an $s-t$ flow in G, denoted by x, that minimizes the dissimilarity between x and x^o by minimizing the support dissimilarity or the total absolute flow deviation, such that we are still able to send a sufficient value of flow from s to t. We will refer to this problem as the biobjective flow problem (BFP).

1.3 Thesis objectives

In this thesis, we investigate various ways to model BFP that are motivated by the goal programming philosophy. Then, we solve, assess, and visualize the quality of solutions produced by the models developed. Thus, our thesis objectives include:

Objective 1 – Modeling BFP: Investigate two ways to model BFP that ensure desired value of flow in the network while minimizing:

1. the support dissimilarity between two solutions, and

2. the total absolute flow deviation between two solutions.

Objective 2 – Implementation and Computational Study: Implement the models from objective 1, and conduct a computational study to investigate the scalability of the models.

Objective 3 – Visualization of solutions: Compare the two models with respect to their optimal solutions by visualizing the optimal solution for both the models in a manner that helps the user understand the merits of each model.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, we provide the necessary background and review some of the existing literature on the maximum flow problem and multi-objective optimization.

2.1 The maximum flow problem

From an algorithmic standpoint, the classical maximum flow algorithms broadly fall into one of the following categories [1].

1. **Augmenting path algorithms:** These algorithms maintain a feasible flow in every iteration and strive to attain optimality by augmenting the flow along an $s-t$ path.

2. **Preflow-push algorithms:** These algorithms violate conservation constraints, allowing nodes to retain excess flow. Such nodes are referred to as *active nodes*. A flow that obeys capacity constraints but violates conservation constraints is called a *preflow*. Preflow-push algorithm then iteratively sends the excess either to the sink or back to the source.

In [4], Ford and Fulkerson presented a labeling algorithm that hinges on the procedure of finding an augmenting $s-t$ path and sending flow along it. This procedure is repeated until no augmenting $s-t$ path to the sink is found and the flow is guaranteed to be maximum according to Theorem 2.

On the other hand, preflow-push algorithms [5] are based on saturating all the source arcs and gradually transferring the excess from an active node to a node nearest

to the sink. In this way preflow-push algorithms get rid of all the excess in the network and the maximum flow is reached.

2.2 Biobjective optimization

Consider a polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$. A linearly constrained multi-objective optimization problem over $\mathcal{P}$ is as shown below,

$$\min_{x \in \mathcal{P}} \quad f_1(x), f_2(x), \ldots f_p(x). \tag{2.1}$$

A biobjective problem is a special case of multi-objective problem and is widely studied. Geoffrion developed a parametric programming approach for solving biobjective optimization problems [6]. A parametric optimization problem given some parameter $t \in [0, 1]$ is stated as follows:

$$\min_{x \in \mathcal{P}} \quad f_t(x) = tf_1(x) + (1-t)f_2(x). \tag{2.2}$$

The optimal solutions to the above parametric problem yield a set of efficient solutions. Furthermore, Lee and Pulat [7], developed an algorithm that modifies the out-of-kilter algorithm [8] to solve the above parametric problem.

Goal programming is another approach extensively used to solve multi-objective optimization problems. The essence of this approach lies in setting target values to each objective in the problem and introducing deviational variables. Then, we minimize the total deviation from the target values. There are many variants that embrace the goal programming philosophy. We list a few of the major approaches next.

1. **Lexicographical goal programming:** This method is generally used when a decision-maker is not able to assign priority weights to the objectives. In such instances, the decision-maker can rank order the objectives. This variant tries to optimize the most important objective first, before optimizing the next important objective over optimal or near optimal solutions to the first objective.

2. **Weighted goal programming:** This method assigns priority weights to the deviation variables that quantify the deviation in the objective from user-specified targets. Thus, weighted goal programming requires directly comparable objectives.

3. **Priotized goal programming:** This approach combines the ideas of lexicographic goal programming and weighted goal programming. Objectives are classified into prioritized groups, and objectives within a group are weighted and aggregated.

In the following we give an example of a goal programming formulation assuming we have p objectives. Here, a_i and c_i denote the priority weights assigned to d_i^- and d_i^+ deviational variables, respectively.

$$\min \quad \sum_{i=1}^{p} (a_i d_i^- + c_i d_i^+) \tag{2.3}$$

subject to:

$$f_1(x) + d_1^- - d_1^+ = b_1 \tag{2.4}$$

$$f_2(x) + d_2^- - d_2^+ = b_2 \tag{2.5}$$

$$\vdots \tag{2.6}$$

$$f_p(x) + d_p^- - d_p^+ = b_p \tag{2.7}$$

$$x \in \mathcal{P} \tag{2.8}$$

$$d_i^+, d_i^- \geq 0 \qquad \forall i = 1 \ldots p \tag{2.9}$$

In the goal programming formulation (2.3) – (2.9), $b_1, \ldots b_p$ are the target values set to each objective and d_i^+, d_i^- are the deviational variables. Thus, we convert the objectives into soft constraints by introducing the deviational variables (d_i^- and d_i^+) as shown above. We also observe from the above formulation that d_i^+ and d_i^- can never simultaneously take positive values in an optimal solution, assuming strictly

positive priorities in the (new) objective function. A more detailed discussion on goal programming can be found in [2].

Many network flow problems are solved using goal programming techniques. Arthur and Lawrence in [9], developed a multi-objective model that aids in making decisions regarding production and shipping over some time intervals. Using goal programming, they assigned targets to the production, shipment and final inventory objectives. Accordingly, priorities were set to each objective, and the model was run using a partitioning algorithm for (linear) goal programming problems (PAGP). PAGP algorithm [10] involves assignment of goal constraints to different priorities and then solving the subproblems involving only the first priority constraints. If alternate optima exist, second priority constraints are included and solved. This procedure is repeated until a unique optimum is found or all goal constraints have been included in the problem. At termination, the current decision variables are locked and all objective values are computed. In [11], Moore and Lee demonstrated an application of goal programming on a transshipment problem, in which cost and labor objectives were considered. They solved it using Lee's modified simplex algorithm [12].

To the best of our knowledge, available literature on biobjective optimization problems does not deal with dissimilarity and flow objectives together. Hence, we would like to examine this problem in the context of flows in networks subject to arc disruptions.

CHAPTER 3

OPTIMIZATION MODELS

We propose optimization models that achieve a sufficient value of flow in G, while minimizing the support dissimilarity or the total absolute flow deviation metric. These models are motivated by the lexicographic goal programming approach. In other words, a decision-maker assigns a flow value to guide our solution. The flow value is controlled by ϵ, such that $0 \leq \epsilon \leq 1$.

3.1 Minimize the support dissimilarity

In this model, we build a model to minimize the support dissimilarity, and obtain a sufficient value of flow in the network. This model uses binary decision variables to trigger arcs in G that minimize the support dissimilarity between x and x^o. We define a binary decision variable y_{ij} for each $(i,j) \in A$ such that,

$$y_{ij} = \begin{cases} 1 & \text{if } x_{ij} > 0, \qquad \forall (i,j) \in A, \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

We note that A can be partitioned into two sets: (i) $A \setminus S(x^o)$, and (ii) $A \cap S(x^o)$. We also know from Section 1.2 that the support dissimilarity is given by $\mid S(x) \setminus S(x^o) \mid + \mid S(x^o) \setminus S(x) \mid$. The binary decision variables, defined in Equation (3.1), help in modeling the sets $S(x) \setminus S(x^o)$ and $S(x^o) \setminus S(x)$. By minimizing $\sum_{(i,j) \in A \setminus S(x^o)} y_{ij}$, we discourage the arcs belonging to the set $A \setminus S(x^o)$ from carrying any flow, and thereby minimize $\mid S(x) \setminus S(x^o) \mid$. Also, the set $S(x^o) \setminus S(x)$ can be written as $(S(x^o) \setminus A) \cup ((S(x^o) \cap A) \setminus S(x))$. The term $\mid (S(x^o) \setminus A) \mid$ is a constant, and can be

ignored in our analysis. Maximizing $\sum_{(i,j)\in A\cap S(x^o)} y_{ij}$ encourages the arcs belonging to the set $A \cap S(x^o)$ to carry flow, resulting in the minimum value of $\mid S(x^o) \setminus S(x) \mid$. Let c^+ and c^- be the weights associated with the arcs in $A \setminus S(x^o)$ and $A \cap S(x^o)$, respectively. In our setting, $c^+ > c^-$, i.e., using the arcs $(i,j) \in S(x) \setminus S(x^o)$ is penalized more than using the arcs $(i,j) \in S(x^o) \setminus S(x)$. With this setting, if we have a perfectly similar solution, the objective function is capable of taking a negative value. So we include the constant term $c^- \mid A \cap S(x^o) \mid$ in the objective function, which provides a lower bound of 0 on the objective value. Therefore, the objective value of 0 indicates a perfectly similar solution. Thus, in the following formulation the objective function minimizes the $\mid S(x) \triangle S(x^o) \mid$.

$$\min \sum_{(i,j)\in A\setminus S(x^o)} c^+ y_{ij} - \sum_{(i,j)\in A\cap S(x^o)} c^- y_{ij} + c^- \mid A \cap S(x^o) \mid \tag{3.2}$$

subject to:

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v, & \text{if} \quad i = s \\ 0, & \forall i \in N \setminus \{s,t\} \\ -v, & \text{if} \quad i = t \end{cases} \tag{3.3}$$

$$v \geq (1-\epsilon)v^* \tag{3.4}$$

$$x_{ij} \leq y_{ij} u_{ij}, \qquad \forall (i,j) \in A \tag{3.5}$$

$$x_{ij} \geq y_{ij}, \qquad \forall (i,j) \in A \tag{3.6}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{3.7}$$

In formulation $(3.2) - (3.7)$, the constraints (3.3) ensure that x is a conserved flow of value v. A sufficient value of flow in the network is obtained via the constraint (3.4), where v^* is the maximum flow in G. If ϵ is 0, we enforce a maximum flow in G. As the ϵ increases, we enforce at least $100(1-\epsilon)\%$ of the maximum flow in G. The constraints (3.5) and (3.6) together capture the rationale presented in Equation

(3.1). If $y_{ij} = 1$, then the constraints (3.5) and (3.6) enforce x_{ij} to lie between 1 and u_{ij}. If $y_{ij} = 0$, then x_{ij} becomes 0. We will henceforth refer to this model as the support dissimilarity model.

3.2 Minimize the total absolute flow deviation

We present a model to find a flow that minimizes the total absolute flow deviation between x and x^o. In formulation (3.8) − (3.10), the constraints (3.9) enforce the flow conservation of value v. Also, the constraints (3.11) provide the lower and upper bound on x_{ij}. The constraint (3.10) is employed to attain a sufficient value of flow in G.

$$\min \sum_{(i,j)\in A} \mid x_{ij} - x^o_{ij} \mid \tag{3.8}$$

subject to:

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v, & \textbf{if} \quad i = s \\ 0, & \forall i \in N \setminus \{s,t\} \\ -v, & \textbf{if} \quad i = t \end{cases} \tag{3.9}$$

$$v \geq (1-\epsilon)v^* \tag{3.10}$$

$$0 \leq x_{ij} \leq u_{ij}, \qquad \forall (i,j) \in A \tag{3.11}$$

The objective function is a piecewise linear convex function. We define a decision variable z_{ij} such that $z_{ij} \geq \max(x_{ij} - x^o_{ij}, -(x_{ij} - x^o_{ij}))\ \forall (i,j) \in A$. The objective function $\mid x_{ij} - x^o_{ij} \mid$ is equal to the smallest value of z_{ij}, for all $(i,j) \in A$ in an optimal solution. In the following model, the constraints (3.13) and (3.14) ensure $z_{ij} \geq \max(x_{ij} - x^o_{ij}, -(x_{ij} - x^o_{ij}))\ \forall (i,j) \in A$.

$$\min \quad \sum_{(i,j)\in A} z_{ij} \tag{3.12}$$

subject to:

$$x_{ij} - x^o_{ij} \leq z_{ij}, \qquad \forall (i,j) \in A \tag{3.13}$$

$$-(x_{ij} - x^o_{ij}) \leq z_{ij}, \qquad \forall (i,j) \in A \tag{3.14}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v, & \textbf{if} \quad i = s \\ 0, & \forall i \in N \setminus \{s,t\} \\ -v, & \textbf{if} \quad i = t \end{cases} \tag{3.15}$$

$$v \geq (1-\epsilon)v^* \tag{3.16}$$

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A \tag{3.17}$$

We will henceforth refer to this model as the total absolute flow deviation model.

3.3 Numerical illustration

We provide a numerical example that shows the trade-off between the dissimilarity metric and the flow obtained. Consider a network G^o with a maximum flow as shown in Figure 3.1. Consider 1 as the source node and 5 as the sink node in G^o.

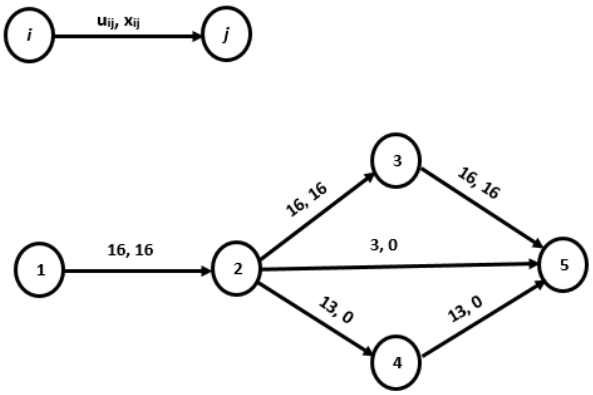

Figure 3.1: Maximum flow in G^o.

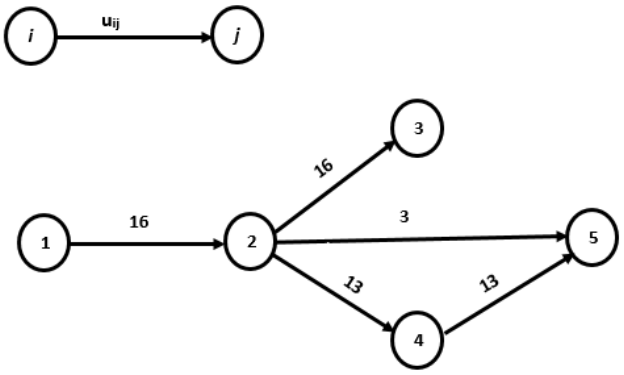

Figure 3.2: G obtained after arc (3,5) is deleted from G^o in Figure 3.1.

Suppose arc $(3, 5)$ undergoes disruption to form a network G as shown in Figure 3.2. We compute the support dissimilarity metric and the flow obtained for various values of ϵ. We set $c^+ = 2$ and $c^- = 1$ to calculate the support dissimilarity objective value. Therefore, the term $c^- \mid A \cap S(x^o) \mid$ in the support dissimilarity objective value is 2.

Let $\epsilon = 0$. We require the maximum flow in G. Figure 3.3 shows a maximum flow of value 16 and the support dissimilarity objective value of 7. The support dissimilarity objective value for the flow in Figure 3.3 is shown in Table 3.1.

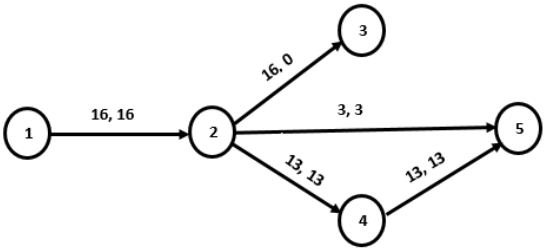

Figure 3.3: Support dissimilarity of 7 when flow obtained is 16.

.

Table 3.1: Support dissimilarity when $\epsilon = 0.0$.

Objective term	**Value**
$\sum_{(i,j)\in A\setminus S(x^o)} c^+ y_{ij}$	$2 \times 3 = 6$
$-(\sum_{(i,j)\in A\cap S(x^o)} c^- y_{ij})$	$-(1 \times 1) = -1$
$c^- \mid A \cap S(x^o) \mid$	$1 \times 2 = 2$
Support dissimilarity	7

Suppose $\epsilon = 0.75$. Then, at least 4 units of flow is enforced by our support dissimilarity model. Figure 3.4 shows 4 units of flow sent along the path $1-2-4-5$. If arc $(2, 5)$ was utilized, then that would have increased the support dissimilarity metric to 7 as seen before. Therefore, to minimize the support dissimilarity objective value, arc $(2, 5)$ is not utilized and all 4 units of flow is sent along path $1-2-4-5$. Table 3.2 summarizes the support dissimilarity observed.

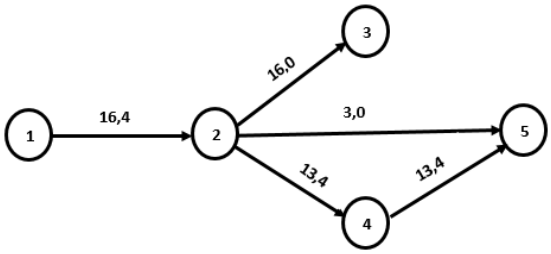

Figure 3.4: Support dissimilarity of 5 when flow obtained is 4.

Table 3.2: Support dissimilarity when $\epsilon = 0.75$.

Objective term	**Value**
$\sum_{(i,j)\in A\setminus S(x^o)} c^+ y_{ij}$	$2 \times 2 = 4$
$-(\sum_{(i,j)\in A\cap S(x^o)} c^- y_{ij})$	$-(1 \times 1) = -1$
$c^- \mid A \cap S(x^o) \mid$	$1 \times 2 = 2$
Support dissimilarity	5

Consider the case when $\epsilon = 1.0$. We can have any flow in G that corresponds

to the minimum support dissimilarity objective value. If the flow is 0, that would correspond to the support dissimilarity objective value of 2. Any flow value greater than 0 would only increase the support dissimilarity objective value. Figure 3.5 shows 0 flow in G. Table 3.3 shows the support dissimilarity objective value for 0 flow.

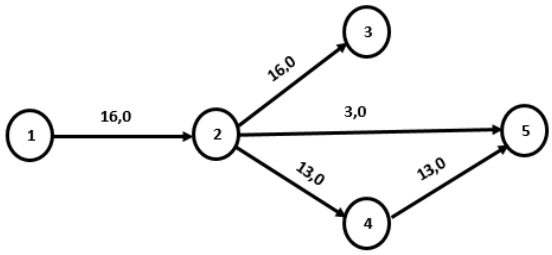

Figure 3.5: Support dissimilarity of 2 when flow obtained is 0.

Table 3.3: Support dissimilarity when $\epsilon = 1.0$.

Objective term	**Value**
$\sum_{(i,j)\in A\setminus S(x^o)} c^+ y_{ij}$	$2 \times 0 = 0$
$-(\sum_{(i,j)\in A\cap S(x^o)} c^- y_{ij})$	$-(1 \times 0) = 0$
$c^- \mid A \cap S(x^o) \mid$	$1 \times 2 = 2$
Support dissimilarity	2

Now, we compute the total absolute flow deviation and the flow obtained for various values of ϵ. Notice that to minimize the total absolute flow deviation, the model must minimize the absolute flow deviation for all $(i, j) \in A$.

When ϵ of 0 is enforced, we obtain the maximum flow as shown in Figure 3.6. The total absolute flow deviation observed for this flow is 45 as shown in Table 3.4.

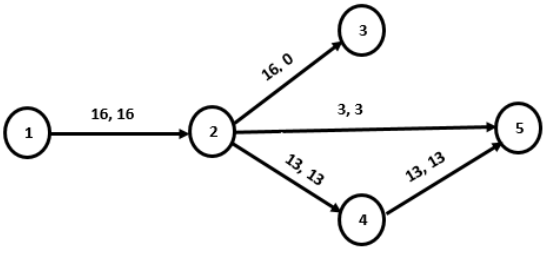

Figure 3.6: Total absolute flow deviation of 45 when flow obtained is 16.

Table 3.4: Total absolute flow deviation when $\epsilon = 0.0$.

$\mid x_{12} - x_{12}^o \mid$	$\mid x_{23} - x_{23}^o \mid$	$\mid x_{24} - x_{24}^o \mid$	$\mid x_{25} - x_{25}^o \mid$	$\mid x_{45} - x_{45}^o \mid$	**Total**
0	16	13	3	13	45

Consider $\epsilon = 0.75$. Then, our model imposes at least 4 units of flow in G. In Figure 3.7, 4 units of flow is sent along arc $(1, 2)$. Three units of flow is sent along arc $(2, 5)$, and one unit of flow is sent along arcs $(2, 4)$ and $(4, 5)$. This flow corresponds to the total absolute flow deviation of 33. Sending all 4 units of flow along path $1 - 2 - 4 - 5$ would increase the total absolute flow deviation metric to 36. The total absolute flow deviation for the flow in Figure 3.7 is computed as shown in Table 3.5.

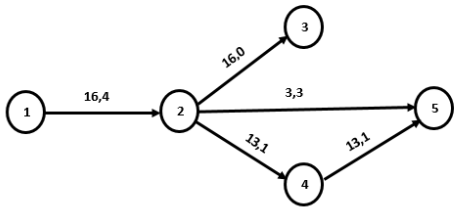

Figure 3.7: Total absolute flow deviation of 39 when flow obtained is 4.

Table 3.5: Total absolute flow deviation when $\epsilon = 0.75$.

$\mid x_{12} - x_{12}^o \mid$	$\mid x_{23} - x_{23}^o \mid$	$\mid x_{24} - x_{24}^o \mid$	$\mid x_{25} - x_{25}^o \mid$	$\mid x_{45} - x_{45}^o \mid$	**Total**
12	16	1	3	1	33

Let $\epsilon = 1.0$. Sending no flow in the network G corresponds to the minimum total absolute flow deviation of 32 as shown in Figure 3.8. Table 3.6 shows the total absolute flow deviation for 0 flow.

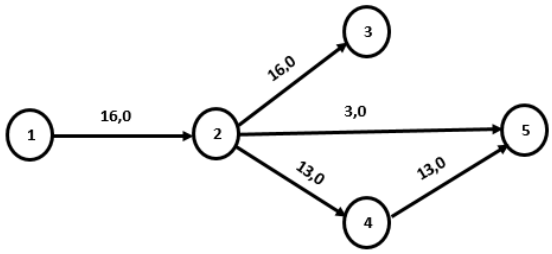

Figure 3.8: Total absolute flow deviation of 0 when flow obtained is 0.

Table 3.6: Total absolute flow deviation when $\epsilon = 1$.

$\mid x_{12} - x_{12}^o \mid$	$\mid x_{23} - x_{23}^o \mid$	$\mid x_{24} - x_{24}^o \mid$	$\mid x_{25} - x_{25}^o \mid$	$\mid x_{45} - x_{45}^o \mid$	**Total**
16	16	0	0	0	32

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this chapter, we discuss the implementation details and the results from solving the models presented in Chapter 3. We solve our models on various testbeds and record the performance of the solver on each model. The purpose of these computational experiments is to investigate the scalability of the models, and understand the impact of the dissimilarity objectives on the solutions obtained. We also visualize the optimal solutions of the models on geospatial data to understand the merits of each model.

4.1 Implementation details and experimental data

We implemented the models of BFP using Python 3.4.3 and Gurobi^{TM} Optimizer 6.5. All the experiments were conducted on 64-bit Windows 10 computer equipped with 8 GB of RAM and Intel® Xeon® 3.20 GHz CPU. The solver parameters were tuned to improve the computational performance of the models without significantly compromising the quality of the optimal solution.

The computational experiments were conducted on the network topologies provided by Zuse Institute Berlin [13]. All instances are directed networks with a source node and a sink node. The arcs were randomly assigned integer capacities from the range $[1, 1000]$. Table 4.1 lists test-instances used to represent G^o. The maximum flow in G^o was computed using NetworkX package (v1.10) of Python.

Table 4.1: Test-instances for network G^o.

Instance	$\mid N \mid$	$\mid A \mid$	v^*	**Time to compute maximum flow (seconds)**
elist96d.rmf	96	528	1931	0.00
elist160d.rmf	160	912	2100	0.02
elist200d.rmf	200	1340	1288	0.02
elist500d.rmf	500	3975	2823	0.08
elist640d.rmf	640	12608	3058	0.20
elist1440d.rmf	1440	22128	4628	0.35
elist2560.rmf	2560	44160	5405	0.72

We invoked the standard implementation of the minimum cut algorithm from Python - NetworkX package to obtain a minimum cut $[S - \bar{S}]$. The instances for G were constructed from G^o using the following procedure: (i) we sorted the arcs in $S(x^o)$ in ascending order of the ratio x^o_{ij}/u^o_{ij}; (ii) then, $\lfloor \frac{|S(x^o)|}{k} \rfloor$ number of top arcs were deleted to generate three networks of G using different values of k. We conducted preliminary experimentation on k, to obtain interesting and nontrivial instances of G that demonstrated trade-off between dissimilarity metrics and flow obtained. The naming convention for the instances of G is the number of nodes followed by a hyphen '-' then the value of k used.

To compute a maximum flow in G, a linear programming formulation for the maximum flow problem was implemented in Gurobi. The presolve level for the support dissimilarity model and the total absolute flow deviation model was set to aggressive. Also, the feasibility tolerance during the experiment was set to the value of 1e-9. That is, the constraint violations were limited to 1e-9.

4.1.1 Model I: Minimizing the support dissimilarity

We tested the model presented in Section 3.1 on the network G. The maximum flow value and the corresponding flow solution for each instance of G were computed before invoking this model. To accelerate the model performance, we provided an initial feasible solution, which was obtained from an optimal flow solution of the maximum flow model. If Gurobi heuristics find a better solution, the model will not be initiated with the feasible solution injected [14]. For the numerical experiments, we set c^+ to 2 and c^- to 1. The root node was solved using three methods: (i) Barrier method, (ii) Dual simplex method, and (iii) Concurrent optimizer. Among these methods, the concurrent optimizer was the fastest method to solve our test instances. Hence, the solver was fixed to run the concurrent optimizer method at the root node. The Gurobi parameter 'IntFeasTol' was set to the minimum value, to increase the numerical accuracy. Table 4.2 summarizes the performance and results of the model for each instance of G. Let RT denote the model solving time and NE denote the number of nodes explored.

We observe from Table 4.2 that as the epsilon value increases from 0 to 0.75, the support dissimilarity objective value decreases in most cases. Also, most of the instances show drastic decrease in the flow obtained as ϵ moves from 0 to 1. Table 4.2 confirms that as the flow obtained in G decreases, the support dissimilarity objective value can either decrease or remain constant. We also observe that the model solving time is maximum for $\epsilon = 0$, where the model is finding the maximum flow.

4.1.2 Model II: Minimizing the total absolute flow deviation

We found from preliminary experiments that the concurrent optimizer was the fastest method to solve the instances of G as before. We present the performance and results of this model for each instance of G, when solved using the concurrent optimization approach in Table 4.3.

Table 4.2: Support dissimilarity model results (Running times reported in seconds).

G	Average time (seconds)		$\epsilon = 0.0$				$\epsilon = 0.75$				$\epsilon = 1.0$			
	Maximum flow	Model building	RT	Obj value	Flow obtained	NE	RT	Obj value	Flow obtained	NE	RT	Obj value	Flow obtained	NE
2560 - 10	6.50	3.33	776.10	30	5405.00	875	4.59	21	1351.25	0	7.87	21	37.00	0
2560 - 15	6.55	3.40	578.53	24	5405.00	823	6.20	18	1351.25	0	4.55	18	38.00	0
2560 - 20	6.59	3.60	8.69	19	5405.00	0	5.25	16	1351.25	0	5.44	16	42.00	0
1440 - 2	2.59	1.98	91818	143	4628.00	60919	9.75	121	1157.00	0	6.48	121	1363.00	0
1440 - 3	2.57	2.16	2.78	87	4628.00	0	2.70	85	2459.00	0	4.01	85	30.00	0
1440 - 4	2.55	2.05	2.28	75	4628.00	0	2.35	75	1157.00	0	2.33	75	851.00	0
160 - 2	0.25	0.37	83.47	111	2059.00	5353	0.45	83	516.00	0	0.53	83	32.00	0
160 - 3	0.39	0.30	0.89	26	2100.00	0	0.52	22	525.00	0	0.40	22	11.00	0
160 - 4	0.33	0.44	0.42	14	2100.00	0	0.55	14	525.00	0	0.27	14	82.00	0
200 - 2	0.23	0.45	79.90	48	1288.00	3545	0.59	27	322.00	0	0.40	27	3.00	0
200 - 3	0.40	0.48	1.29	27	1288.00	0	0.52	21	322.00	0	0.48	21	8.00	0
200 - 4	0.25	0.49	3.17	24	1288.00	69	0.60	18	322.00	0	0.52	18	318.00	0
500 - 2	0.42	0.31	85669	79	2823.00	1057899	0.12	51	1312.00	0	0.93	51	1276.00	0
500 - 3	0.40	0.33	12.13	20	2823.00	1063	0.53	16	1789.00	0	0.58	16	1665.00	0
500 - 4	0.41	0.37	0.62	4	2823.00	0	0.80	4	2579.00	0	0.65	4	2393.00	0
640 - 2	0.90	0.98	45.59	79	3058.00	0	2.40	70	1260.00	0	2.01	70	1219.00	0
640 - 3	1.10	1.16	2.39	46	3058.00	0	1.99	45	2228.00	0	1.99	45	1967.00	0
640 - 4	1.27	1.21	2.18	48	3058.00	0	1.72	47	2178.00	0	1.50	47	1906.00	0
96 - 2	0.25	0.30	23.48	57	1539.00	2679	0.25	32	384.75	0	0.22	32	53.00	0
96 - 3	0.27	0.27	0.69	42	1841.00	0	0.20	25	460.25	0	0.43	25	10.00	0
96 - 4	0.28	0.28	0.58	26	1931.00	0	0.17	19	482.75	0	0.47	19	9.00	0

Table 4.3: Total absolute flow deviation model results (Running times reported in seconds).

G	Average time (seconds)		$\epsilon = 0.0$			$\epsilon = 0.75$			$\epsilon = 1.0$		
	Maximum flow	Model building	RT	Obj value	Flow obtained	RT	Obj value	Flow obtained	RT	Obj value	Flow obtained
2560 - 10	6.80	4.93	2.58	318	5405	0.75	261	5384	0.72	261	5384
2560 - 15	6.65	5.38	1.82	151	5405	0.72	142	5394	0.70	142	5394
2560 - 20	6.85	5.19	1.11	98	5405	0.72	86	5397	0.69	86	5397
1440 - 2	2.59	2.53	0.7	16565	4628	0.63	15906	4296	0.49	15906	4296
1440 - 3	2.53	2.56	0.55	7044	4628	0.43	6733	4453	0.43	6733	4453
1440 - 4	2.51	2.59	0.71	5468	4628	0.43	5319	4517	0.39	5319	4517
160 - 2	0.20	0.25	0.05	11438	2059	0.13	9953	1607	0.03	9953	1607
160 - 3	0.24	0.3	0.13	5298	2100	0.05	5213	1997	0.04	5213	1997
160 - 4	0.42	0.22	0.18	2995	2100	0.15	2952	2058	0.04	2952	2058
200 - 2	0.50	0.39	0.05	5103	1288	0.08	4769	1088	0.03	4769	1088
200 - 3	0.29	0.15	0.07	3075	1288	0.05	2997	1217	0.06	2997	1217
200 - 4	0.17	0.17	0.06	2079	1288	0.05	2028	1226	0.05	2028	1226
500 - 2	0.4	0.42	0.2	16286	2823	0.13	15768	2568	0.19	15768	2568
500 - 3	0.64	0.55	0.22	9382	2823	0.15	9382	2823	0.26	9382	2823
500 - 4	0.6	0.53	0.17	5723	2823	0.15	5723	2823	0.15	5723	2823
640 - 2	1.12	1.44	0.5	13833	3058	0.49	13684	2897	0.40	13684	2897
640 - 3	1.23	1.43	0.37	6903	3058	0.36	6843	2974	0.40	6843	2974
640 - 4	1.26	1.6	0.27	4977	3058	0.27	4977	3036	0.38	4977	3036
96 - 2	0.24	0.2	0.12	8959	1539	0.03	7331	876	0.03	7331	876
96 - 3	0.23	0.12	0.05	6032	1841	0.11	4838	1321	0.08	4838	1321
96 - 4	0.10	0.07	0.12	3301	1931	0.11	3116	1790	0.03	3116	1790

From Table 4.3, we observe that as the epsilon value increases from 0 to 0.75, the total absolute flow deviation metric and the flow obtained decrease. We also observe significant decrease in the model solving time as the epsilon value increases from 0 to 0.75. The total absolute flow deviation and the flow obtained has remained the same for $\epsilon = 0.75$ and $\epsilon = 1$. Therefore, we conclude that as the flow obtained decreases, the total absolute flow deviation may either decrease or stay constant.

4.2 Visualization case studies

Network visualization utilizes the perceptive ability of humans to better understand the problem setting and flow in networks. Apart from being visually appealing, it facilitates exploratory analysis of network data, and thus leads a user through a complex analytical process. We used the network visualization to convey to the end-user the model solution and the associated dissimilarities. This helps the end-user to easily understand the dissimilarity between x and x^o, and thereby assess the behavior of our models. Thus, visualization enables the end-user to assess the network disruption with much more clarity. We performed all the visualizations with the open source software Gephi 0.9 [15].

We adopted the geospatial dataset from TranStats - Bureau of Transportation [16] to visualize our model solution. The dataset was manipulated to obtain 94 cities from the mainland United States and 207 arcs joining them. The cities in the dataset represented nodes of network G^o and their coordinates were available. The arc structure was amended to obtain the density of the network that would facilitate effortless visualization. The arcs were assigned with a random integer capacity ranging from $[1, 1000]$. This network was represented as G^o, with a source node represented by 'Hartford CT' and a sink node represented by 'Kansas City MO'. We assigned a random number between [0,1] to each arc in G^o, and chose to delete the top 20% of

the arcs to construct G. Thus, G comprises of 94 nodes and 194 arcs. The source node and the sink node remained the same.

4.2.1 Model I: Minimizing the support dissimilarity

We considered a hypothetical situation, where there was already a flow of maximum value from 'Hartford CT' to 'Kansas City MO' in G^o. The following figure 4.1 illustrates such maximum flow, along with the arcs susceptible to disruption, indicated in red. Thick arcs indicate $S(x^o)$, while thin arcs indicate $A^o \setminus S(x^o)$.

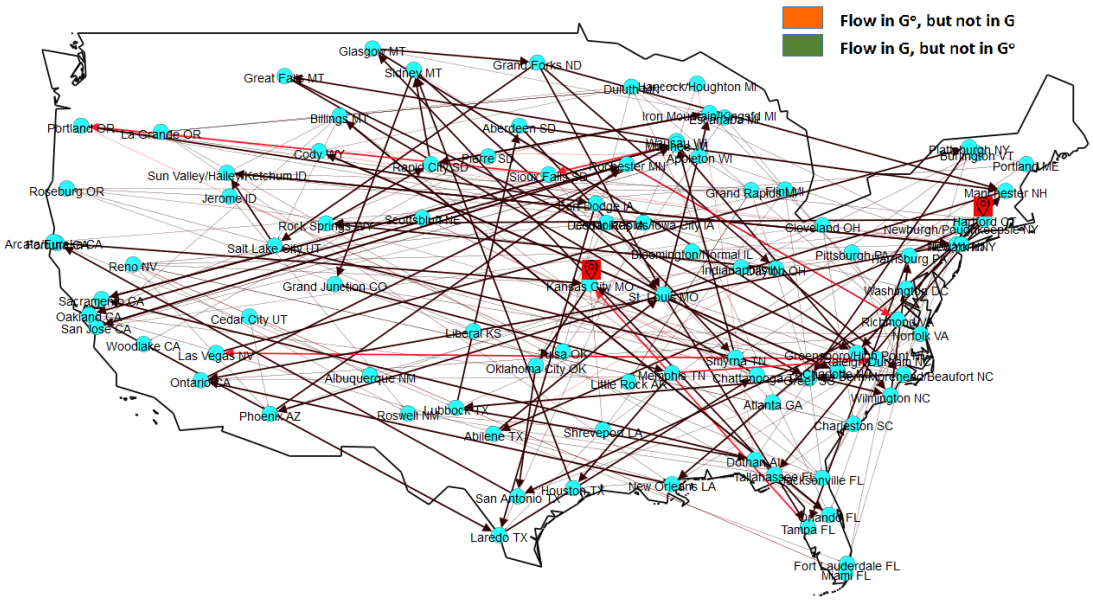

Figure 4.1: Maximum flow in network G^o.

The network G^o underwent a disruption to form a network G, with an arc set denoted by A. We solved a maximum flow model on this network G to obtain a flow of the maximum value. The associated support dissimilarity between x and x^o is shown in Figure 4.2.

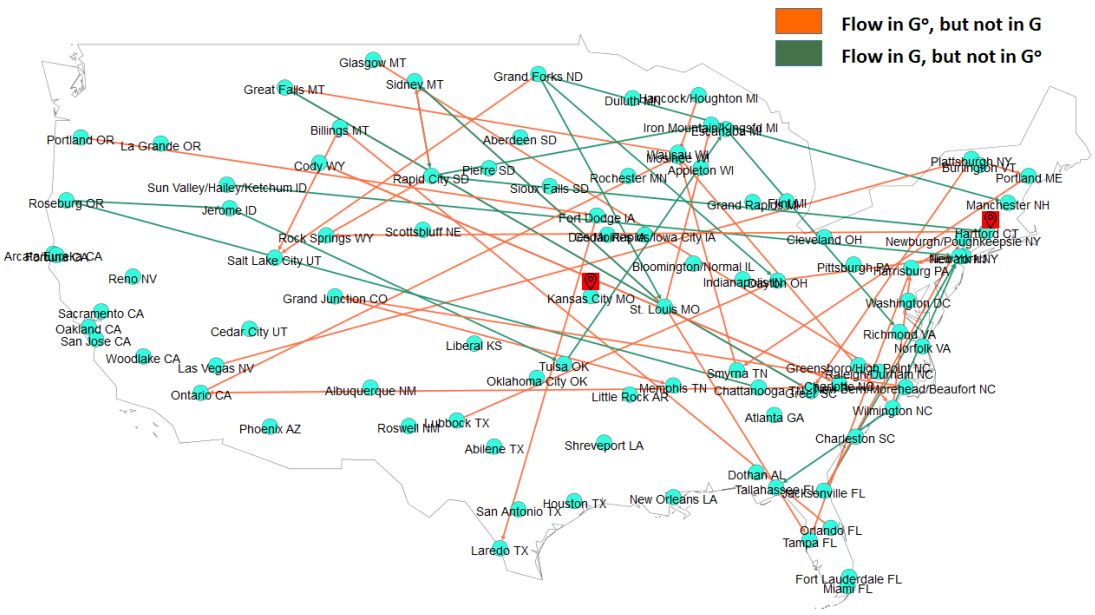

Figure 4.2: Support dissimilarity in network G using the maximum flow model.

There are two types of arcs contributing to the dissimilarity metric: (i) Arcs belonging to $S(x^o)$, but not to $S(x)$, and (ii) arcs belonging to $S(x)$, but not to $S(x^o)$. The following table summarizes the count of these two types of arcs observed in Figure 4.2:

Table 4.4: Support dissimilarity in network G using the maximum flow model.

Arcs adding to the support dissimilarity	**Count**
Flow in G^o, but not in G	33
Flow in G, but not in G^o	18
Total arcs	51

In situations where we would like to adopt the original plan, the above model solutions could prove costly. Therefore, to minimize the support dissimilarity metric, we solved the support dissimilarity model on G. In order to obtain the maximum flow in G, the epsilon value was set to 0. Also, c^+ and c^- were set to 2 and 1, respectively.

We clearly see fewer arcs in Figure 4.3 as compared to Figure 4.2, indicating less dissimilarity between x and x^o. Since $c^+ > c^-$, we observe far fewer green arcs than

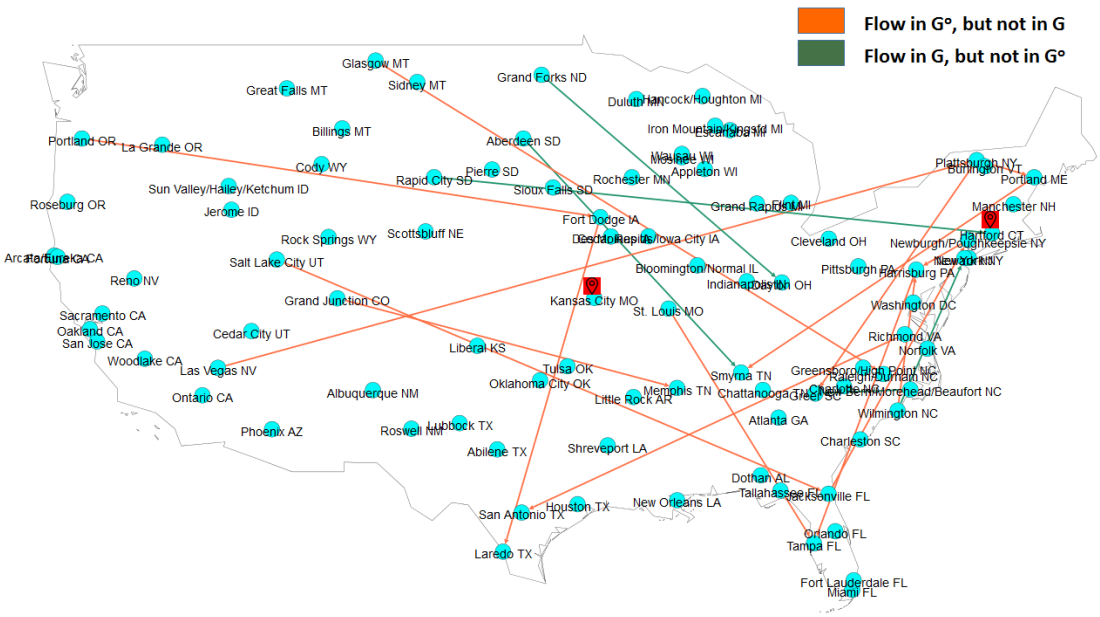

Figure 4.3: Support dissimilarity in network G using the support dissimilarity model.

the red arcs. As seen from Table 4.5, our model utilizes fewer arcs which belong to $A \setminus S(x^o)$.

Table 4.5: Support dissimilarity in network G using the support dissimilarity model.

Arcs adding to the support dissimilarity	**Count**
Flow in G^o, but not in G	15
Flow in G, but not in G^o	5
Total dissimilar arcs	20

We clearly observe that the number of arcs contributing to the support dissimilarity has halved. As the ϵ was set to 0, the model enforced the maximum flow in G. In this manner, our model allows us to minimally change the original operating plans.

4.2.2 Model II: Minimizing the total absolute flow deviation

In this section, we compared the total absolute flow deviation with the maximum flow model. We used the thickness of the arcs to convey the amount of flow on the arc.

Absolute flow deviation between x_{ij} and x^o_{ij} is depicted by the color of the arc. The color scheme adopted in the visualization of the total absolute flow deviation is based on [17]. Figure 4.4 shows the arcs contributing to the total absolute flow deviation metric in G for the maximum flow model. Table 4.6 summarizes the number of arcs observed in Figure 4.4.

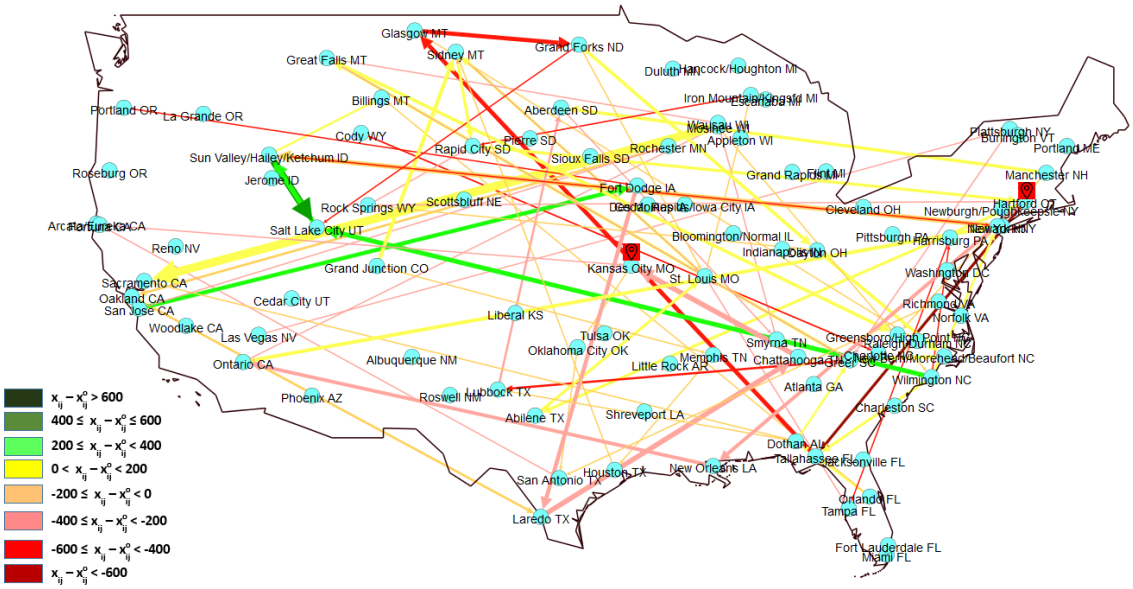

Figure 4.4: Total absolute flow deviation in network G using the maximum flow model.

Table 4.6: Total absolute flow deviation in network G using the maximum flow model.

Arcs adding to Total Absolute Flow Deviation	**Count**
$x_{ij} - x^o_{ij} > 600$	0
$400 \leq x_{ij} - x^o_{ij} \leq 600$	1
$200 \leq x_{ij} - x^o_{ij} < 400$	3
$0 < x_{ij} - x^o_{ij} < 200$	19
$-200 \leq x_{ij} - x^o_{ij} < 0$	19
$-400 \leq x_{ij} - x^o_{ij} < -200$	22
$-600 \leq x_{ij} - x^o_{ij} < -400$	11
$x_{ij} - x^o_{ij} < -600$	1
Total arcs	76

To minimize the absolute difference between x and x^o, we solved the total absolute

flow deviation model on G. The epsilon value of 0 was set to obtain the maximum flow. Figure 4.5 shows the arcs contributing to the total absolute flow deviation metric using the total absolute flow deviation model. Table 4.7 summarizes the various arcs observed in Figure 4.5.

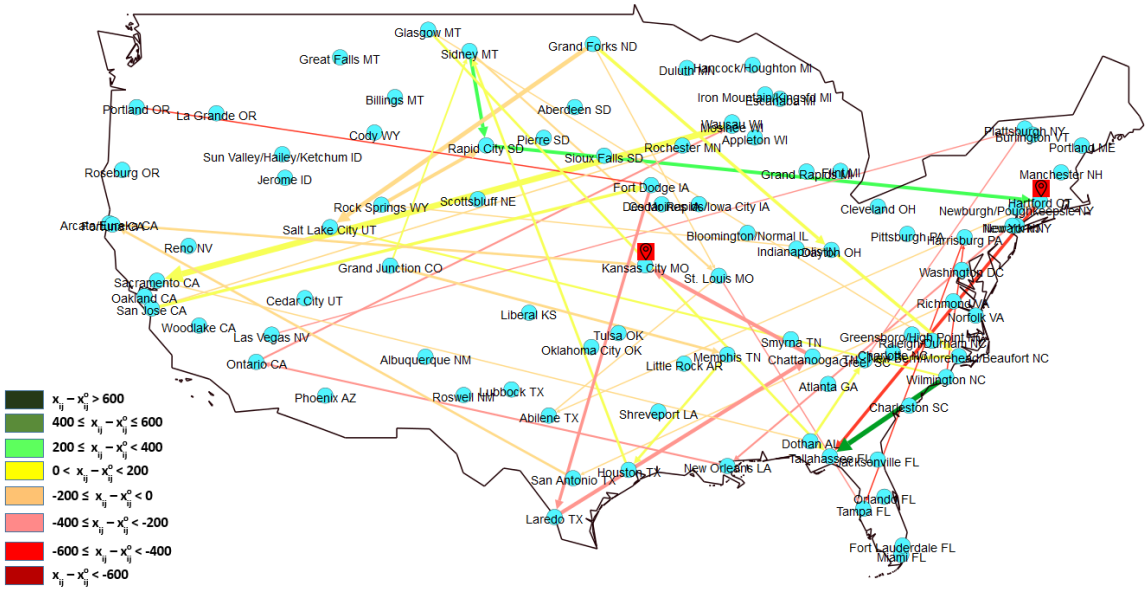

Figure 4.5: Total absolute flow deviation in network G using the total absolute flow deviation model.

Table 4.7: Total absolute flow deviation in network G using the total absolute flow deviation model.

Arcs adding to Total Absolute Flow Deviation	**Count**
$x_{ij} - x^o_{ij} > 600$	0
$400 \leq x_{ij} - x^o_{ij} \leq 600$	1
$200 \leq x_{ij} - x^o_{ij} < 400$	2
$0 < x_{ij} - x^o_{ij} < 200$	11
$-200 \leq x_{ij} - x^o_{ij} < 0$	14
$-400 \leq x_{ij} - x^o_{ij} < -200$	9
$-600 \leq x_{ij} - x^o_{ij} < -400$	4
$x_{ij} - x^o_{ij} < -600$	0
Total arcs	41

From Tables 4.6 and 4.7, we see a drastic decrease in the number of arcs whose absolute flow deviations are greater than 200. Thus, our model not only obtained the maximum flow, but also decreased the total absolute flow deviation between x and x^o.

4.3 Impact of number of disrupted arcs on dissimilarity

We further conducted a visualization experiment to understand the impact of number of disrupted edges on the dissimilarity metrics. This experiment is intended to guide the end-user through the post-optimization analysis involving links prone to disruption.

We worked on the same network G^o as mentioned in Section 4.2. The arcs prone to disruption were ordered according to x^o_{ij}/u_{ij}. These arcs were then deleted in three steps. In the first step, we deleted top 4 arcs to build a network G^{1-4}. In the second step, the next 4 arcs were deleted to build a network G^{1-8}. And, in the final step, all the arcs prone to disruption were deleted (this is G). We ran the maximum flow model on each of these networks G^{1-4}, G^{1-8}, G, and they all yielded the same maximum flow value. To understand the impact of number of disrupted arcs on the support dissimilarity, we deployed our models on G^{1-4}, G^{1-8}, and G. The following visualizations in Figures 4.6 and 4.7 are the results of our models on G^{1-4} and G^{1-8}, respectively. The results for G are already shown in Figure 4.3.

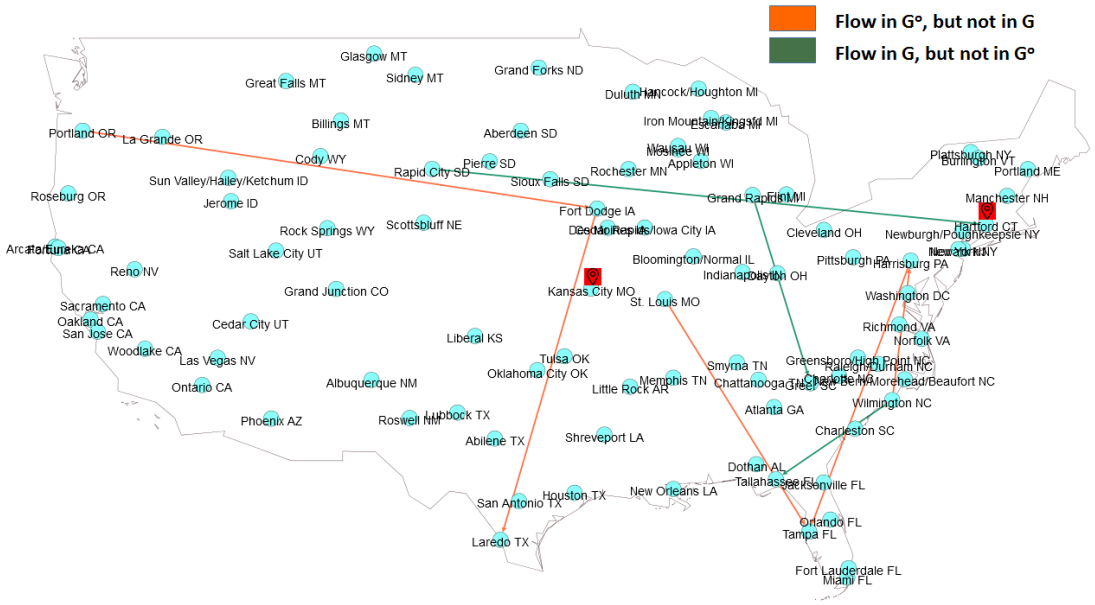

Figure 4.6: Support dissimilarity in network G^{1-4}.

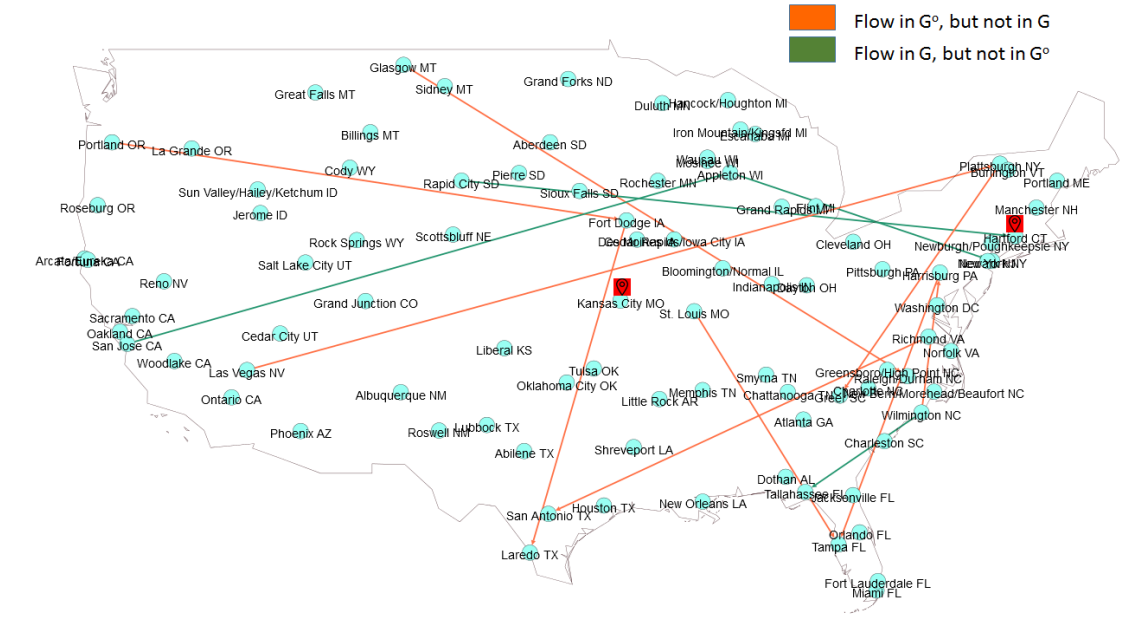

Figure 4.7: Support dissimilarity in network G^{1-8}.

We observe fewer dissimilar arcs in Figure 4.6 as compared to Figure 4.7. Figure 4.3 exhibits the most number of arcs among networks G^{1-4}, G^{1-8}, and G. We can therefore conclude from Figures 4.3, 4.6, and 4.7 that as the number of disrupted arcs increases, the support dissimilarity increases.

Next, we visualize the impact on the total absolute flow deviation metric. The following visualizations in Figures 4.8 and 4.9 resulted from solving the total absolute flow deviation model on G^{1-4} and G^{1-8}, respectively.

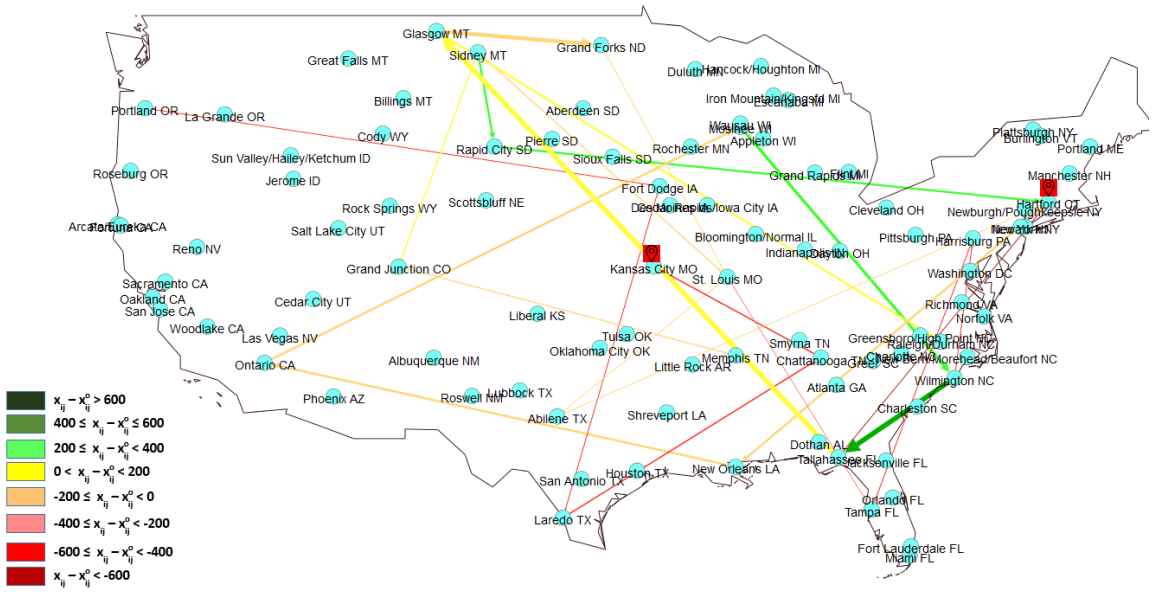

Figure 4.8: Total absolute flow deviation in network G^{1-4}.

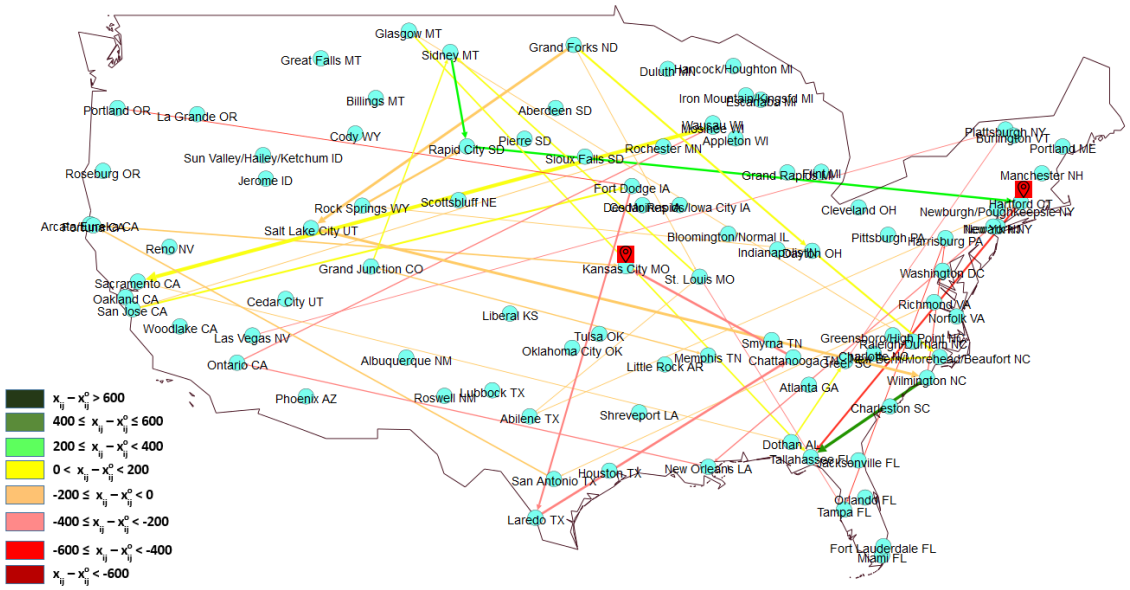

Figure 4.9: Total absolute flow deviation in network G^{1-8}.

The result for the network G was presented in Figure 4.5. From Figures 4.5, 4.8, and 4.9, we conclude that as the number of disrupted arcs increases, the total absolute flow deviation metric can either increase or stay constant.

Thus, visualization helped us to observe the impact of the disrupted arcs on the dissimilarity metrics.

CHAPTER 5

CONCLUDING COMMENTS AND FUTURE WORK

Transportation and supply chain networks are witnessing disruptions with increasing frequency [18]. Disruptions can not only impact the capacity of the links to carry the flow, but also breach the commitments made to suppliers, distribution partners, and customers. Therefore, it is necessary to have contingency plans in place. In this thesis, we developed models that enable a decision maker to address network disruptions. The models not only attain the desired flow but also support the commitments made before disruption.

This thesis defined two metrics, namely support dissimilarity and total absolute flow deviation that measure the dissimilarity between the flow before disruption and the flow after disruption. We developed models to minimize the support dissimilarity and the total absolute flow deviation between two flows. We adopted the best implementation of these models on Python and Gurobi by tuning parameters that improved the computational performance as well as resolved numerical issues. To convey our model solutions to the end-user, we developed visualizations that provided intuition of the dissimilarity metrics and the associated model solutions. The visualizations clearly showed the merits of our models over solving a maximum flow model, by helping us visualize our model solutions against the maximum flow solution. We also used visualization to understand the impacts of number of disrupted edges on the dissimilarity metrics.

One possible area of future research is to incorporate probabilistic arc failures, and model it in a stochastic optimization setting. This setup would enable us to address disruptions beforehand with much more clarity. We can still improve the performance of total absolute flow deviation model, if we can find faster algorithms for solving linear programming models with piecewise convex objective function. Other future work includes determining the computational complexity of the support dissimilarity model.

From the visualization perspective, we worked on a relatively small dataset that allowed clear visualization. It would be interesting to visualize a 3D network model that would allow us to fit a network of larger size, without aggregation of information, and also allowing cleaner visualizations. Another important area of research in visualization, would be allowing the end-user to participate in tweaking or modifying the optimal solution based on other considerations.

REFERENCES

[1] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms and Applications*. Englewood Cliffs, New Jersey: Prentice Hall, 1993.

[2] D. Jones and M. Tamiz, *Practical Goal Programming*. New York, US: Springer, 2010.

[3] R. Rardin, *Optimization in Operations Research*. Englewood Cliffs, New Jersey: Prentice Hall, 1998.

[4] L. R. Ford Jr and D. R. Fulkerson, "A simple algorithm for finding maximal network flows and an application to the hitchcock problem," tech. rep., DTIC Document, 1955.

[5] A. Goldberg and R. Tarjan, "A new approach to max flow," *Journal of ACM*, vol. 35, no. 4, 2008.

[6] A. Geoffrion, "Solving bicriterion mathematical programs," *Operations Research*, vol. 15, no. 1, pp. 39–54, 1967.

[7] H. Lee and S. Pulat, "Bicriteria network flow problems: Continuous case," *European Journal of Operations Research*, vol. 51, no. 1, pp. 119–126, 1991.

[8] D. Fulkerson, "An Out-of-Kilter Method for Minimal-Cost Flow Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 1, pp. 18–27, 1961.

[9] L. Arthur and K. Lawrence, "Multiple goal production and logistics planning in a chemical and pharmaceutical company," *Computers and Operations research*, vol. 9, no. 2, pp. 127–137, 1982.

[10] J. Arthur and A. Ravindran, "PAGP, a partitioning algorithm for (linear) goal programming problems," *ACM Transactions on Mathematical Software*, vol. 6, no. 3, pp. 378–386, 1980.

[11] L. Moore, B. Taylor III, and S. Lee, "Analysis of a transhipment problem with multiple conflicting objectives," *Computers and Operations research*, vol. 5, no. 1, pp. 39–46, 1978.

[12] S. M. Lee *et al.*, *Goal programming for decision analysis*. Philadelphia, US: Auerbach, 1972.

[13] G. Skorobohatyj, "Maximum flow problem instances." `http://elib.zib.de/pub/Packages/mp-testdata/maxflow/`.

[14] Gurobi Optimization, Inc., "Gurobi optimizer reference manual, version 6.5, copyright © 2015." `http://www.gurobi.com/documentation/6.0/refman/`, 2016.

[15] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009.

[16] U.S. Department of Transportation, Research and Innovative Technology Administration, Bureau of Transportation Statistics, Freight Transportation: Global Highlights, 2010. `http://www.bts.gov/`. Accessed May 2016.

[17] Brewer A. C, Penn State. `www.ColorBrewer.org`. Accessed Nov 2016.

[18] C. Bode, *Causes and effects of supply chain disruptions*. PhD thesis, WHU–Otto Beisheim School of Management, 2008.

[19] D. Altner and O. Ergun, "Rapidly solving an online sequence of maximum flow problems," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (L. Michel, ed.), (Spain), pp. 283–287, Springer, 2008.

VITA

Sampreet Sudheer Mangalvedhe

Candidate for the Degree of

Master of Science

Thesis: ON A BIOBJECTIVE FLOW PROBLEM IN NETWORKS

Major Field: Industrial Engineering & Management

Biographical:

Personal Data: Born in Miraj, Maharashtra, India on September 29, 1990.

Education:

Received the B.E. degree from University of Mumbai, Mumbai, Maharashtra, India, 2012, in Mechanical Engineering

Completed the requirements for the degree of Master of Science with a major in Industrial Engineering & Management Oklahoma State University in December, 2016.

Experience:

Graduate Teaching Assitant - School of Industrial Engineering & Management, Oklahoma State University (August 2015 - December 2016)

Graduate Trainee Engineer - Tenova, India (July 2012 - June 2013)